Direct Numerical Simulation of Turbulent Flows on Roadrunner

Jamaludin Mohd-Yusof, Daniel Livescu, Mark R. Petersen, CCS-2;
Nehal Desai, CCS-1

Turbulence is an inherently multiscale phenomenon, encompassing a large range of length and time scales, all of which must be resolved in a direct numerical simulation (DNS). The technique requires that the numerical errors remain small and not affect the solution. DNS allows a degree of control of specific physical phenomena not accessible in experiments, leading to improved understanding and, ultimately, to models for large multiphysics codes. However, the numerical conditions significantly restrict the range of flows and parameters accessible, as they require very large grids. The current progress in computer hardware, based on heterogeneous architectures, could significantly advance our state-of-the-art turbulence simulations and, thus, turbulence theory and modeling. The IBM Cell Broadband Engine (Cell BE) used in the Roadrunner computer is an example of the newest generation of accelerators. Common to many of these architectures is the existence of multiple processing cores (synergistic processing units, or SPUs, in this case) with independent, high-bandwidth access to local memory.

CFDNS is a Fortran-structured grid code that simulates the compressible Navier-Stokes equations in 3D. Multiple species, each with realistic material properties equation of state (EOS) are allowed, as well as Cartesian, cylindrical, and spherical grid geometries. The bulk of the time in the code is spent in either computing the update equations or computing the derivatives of the various fields. The update equation is computed in a pointwise manner throughout the domain and is straightforward to vectorize once all the terms have been evaluated. The derivative step uses a compact finite difference (Pade) scheme that requires a tridiagonal inversion along the direction in which the derivative is being performed. Since the data is laid out in 3D, this requires a strided load of the data (Fig. 1).

A subset of the base code was ported to the Cell BE architecture and implemented in C with SPU extensions. At present this version only allows periodic boundary conditions, with ideal gas EOS. The derivative calculation vectorizes naturally in the x- and y- directions, since the data is contiguous in z. For the z-derivative, the data must be reordered in local store to achieve good performance, but the data is not reordered in main memory. Each SPU works on a different portion of the data and executes its own direct memory access (DMA) calls independently, to maximize performance. Similar data partitioning is used for the update step.

The Cell BE implementation of the code was benchmarked against a functionally identical C code running on the Opteron. Significant speedups of 40x (64^3) to 50x (128^3) were observed. The primary benefit for the Cell BE implementation is the ability to independently access main memory, allowing an effective memory access rate of $\sim$12 GB/s (50 percent of theoretical peak) [1]. Further optimization is in progress.

A full parallel implementation of the code is in progress, in anticipation of the full Roadrunner deployment, which will allow simulations at resolutions of 4096^3. These will be by far the largest compressible turbulence simulations ever performed, and we expect them to allow significant advancement on the theory and modeling of compressible turbulence.

For more information contact Jamaludin Mohd-Yusof at jamal@lanl.gov.

[1] J. Mohd-Yusof, N. Desai, D. Livescu, "Roadrunner: Compressible Turbulence Simulation," presented at SC07 (International Conference for High Performance Computing, Network, Storage and Analysis) (2007).

Funding Acknowledgments
- Department of Energy, National Nuclear Security Administration
- Department of Energy, National Nuclear Security Administration, Advanced Simulation and Computing Program
- Los Alamos National Laboratory Directed Research and Development Program

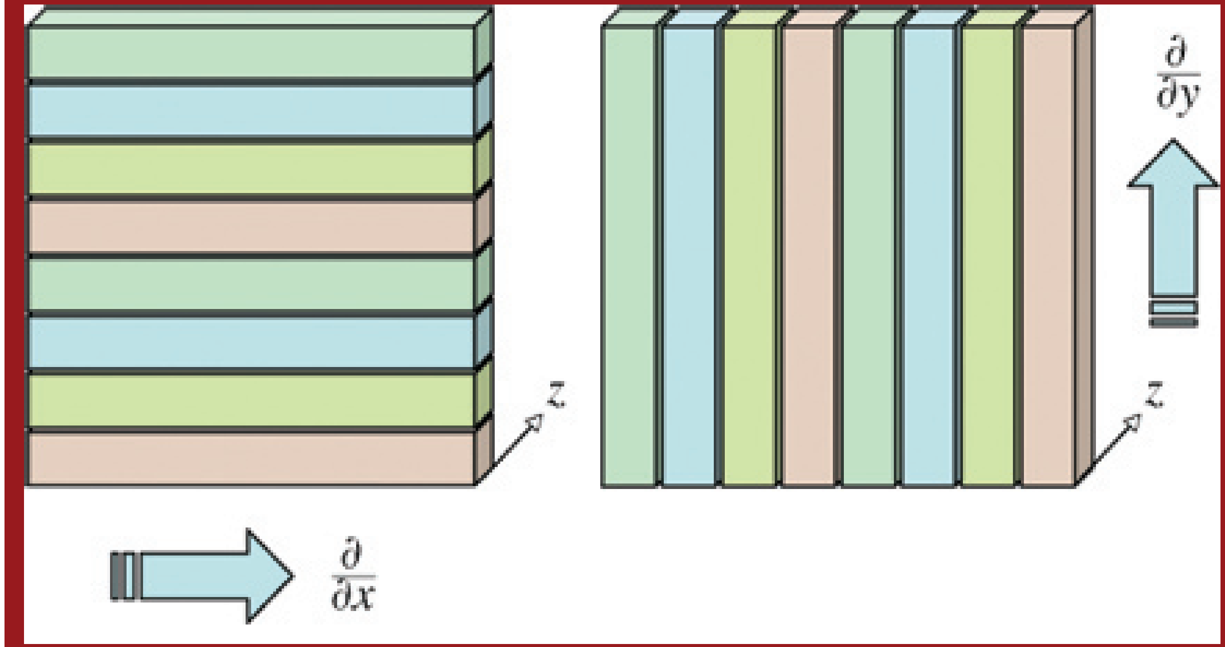

Fig. 1. Depiction of the division of work between SPUs for the x- and y- derivative calculation. Since each derivative requires data along the entire line in the direction of the derivative being taken, different partitioning is required for each direction. The z-direction derivative and the update equation can use either partitioning.